Two-dimensional Ising model Monte Carlo simulation

R.L. Carnielli - RA 157139

8th August 2020

This document is an assignment of the module F 604 - Statistical Physics offered by "Gleb Wataghin" Institute of Physics - University of Campinas. In the following sections, there is a simple demonstration of the Monte Carlo simulation. It agrees with analytical expressions of the two-dimensional Ising model. Moreover, there is also a simplified relationship investigation of the lattice size and the specific heat capacity at low temperatures.

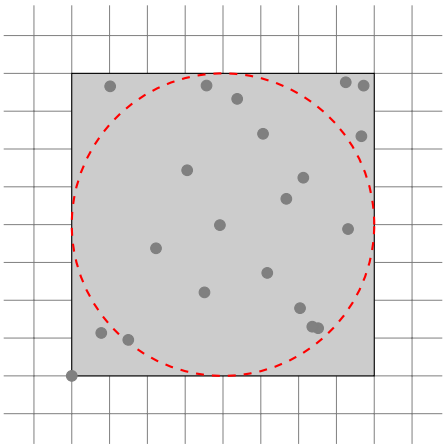
1. Introduction

Statistical physics aims to explain the behavior and the evolution of physical systems which have a large number of particles (called macroscopic systems), from the characteristics of their microscopic constituents.
The results presented herein use the Monte Carlo method to describe a two-dimensional ferromagnetic system. The Monte Carlo Method is based on a statistical approach to compute the time evolution of the system from a certain initial condition. Based on heuristic arguments, the Monte Carlo method enables the formulation of the so-called Metropolis algorithm that will be used to compute the time evolution of a system. Its robustness is demonstrated by numerical simulations that capture the phase transition of a two-dimensional ferromagnetic systems with no external magnetic field $H = 0$. The microscopic interaction between spins are based on the Ising model and it allows to model the behavior of macroscopic phenomena such as the phase transition. Also it is possible to calculate macroscopic physical quantities such as the crystal magnetization.

2. Monte Carlo Method

Monte Carlo method's major idea is to compute deterministic quantities through stochastic phenomena. The classic Monte Carlo's example consists in computing an approximate value of π. Suppose there is a square panel of side 2 with a delimited circle of radius 1 inside of it (see the figure below). Suppose one can choose randomly N positions (x,y) on the panel.
The ratio R between the number of positions inside the circle N_c and the total number of choices N is an approximate value for $\pi/4$. In the limit $N \to \infty$ the ratio goes to $\pi/4$.

$$R = \lim_{N\to\infty} \frac{N_c}{N} = \frac{\pi}{4} \tag{2.1}$$



In a similar way, an stochastic method was developed in Metropolis et al. [1953] in order to compute the time evolution of a system of interacting molecules. This model is based on the *canonical ensemble* framework which supposes the system to be on a fixed temperature. It means that there is a heat bath connected to the system in order to provide the right amount of energy and maintain it at a fixed temperature. In the section "Simulations of the canonical ensemble" of Gould and Tobochnik [2010], the author describes the so-called Metropolis algorithm in a easy-to-understand way:

> "The method is based on the fact that the ratio of the probability that the system is in state j with energy E_j to the probability of being in state i with energy E_i is $p_j/p_i = e^{-\beta(E_j - E_i)} = e^{-\beta\Delta E}$, where $\Delta E = Ej - Ei$ and $\beta = 1/K_bT$. We then interpret this ratio as the probability of making a transition from state i to state j. If $\Delta E < 0$, the quantity $e^{-\beta\Delta E}$ is greater than unity, and the probability is unity."

In order to make a time-step evolution given E_j and E_i you should: compute ΔE, if $\Delta E < 0$, you accept the transition $i \to j$. Otherwise you will accept the transition with a probability $e^{-\beta\Delta E}$. The algorithm described above will be discussed in more details for the 2D Ising model in the section 3.2.

3. Ising model

Insing's hamiltonian H_{ising} (equation 3.1) proposed by Ising, represents a lattice in which the spins are only allowed to be in the $\hat{\mathbf{z}}$ direction. This model can be interpreted as a limit of Heisenberg's hamiltonian in which the spins are allowed to be in any direction. This kind of hamiltonian is composed by two main terms: the first term represents the interaction between spins and the second term represents the interaction of the spins with an external magnetic field. The Exchange Interaction J for a ferromagnetic system is a positive constant - and is negative for an anti-ferromagnetic system. The Spin σ_i is the i-th spin value of each lattice cell which can be $+1$ or -1. The external Magnetic Field is $\mathbf{H} = \mathrm{H}\hat{\mathbf{z}}$.

$$H_{ising} = -\mathrm{J}\sum_{\langle i,j\rangle}\sigma_i\,\sigma_j - \mathrm{H}\sum_{i=1}\sigma_i \tag{3.1}$$

The sum on $\langle i, j\rangle$ refers to the product of the i-cell times its j first neighbors. In a 2-dimensional square lattice there are 4 neighbors for each cell except for those on the edges. One can notice the equation 3.1 with $\mathrm{J} > 0$ has a low energy state whenever the lattice's spins are all up ($\sigma_i = +1\ \forall i$) or all down ($\sigma_i = -1\ \forall i$). From Gould and Tobochnik [2010], the energy per particle (equation 3.3), the magnetization (equation 3.2), the magnetic susceptibility (equation

3.4) and the specific heat capacity (equation 3.5) can be written as a function of the spins σ_i and the hamiltonian H_{ising}. These equations are valid for a system even with a finite number of particles and, for the susceptibility and the heat capacity, it is required the system to be in equilibrium.

$$m = \frac{M}{N} = \frac{1}{N}\sum_i \sigma_i \tag{3.2}$$

$$e = \frac{E}{N} = -\frac{\mathrm{J}}{N}\sum_{\langle i,j\rangle} \sigma_i\,\sigma_j \tag{3.3}$$

$$\chi = \frac{1}{k_B T}\left(\left\langle M^2\right\rangle - \langle M\rangle^2\right) \tag{3.4}$$

$$C = \frac{1}{k_B T^2}\left(\left\langle E^2\right\rangle - \langle E\rangle^2\right) \tag{3.5}$$

The Ising model has a phase transition between an ordered and a disordered phase in 2D or 3D. The simulations presented in the next sections have a transient regime preceding the equilibrium. Therefore, in order to compute the physical quantities using equations 3.2, 3.3, 3.4 and 3.5, the system must be in equilibrium.

3.1. Order-disorder transition of two-dimensional Ising model

In Yang [1952] and Onsager [1944] is presented analytic expressions for physical quantities of the two-dimensional Ising model. Considering a square lattice laying on a $\hat{\mathbf{x}}$-$\hat{\mathbf{y}}$ plane. The spins are allowed to be only in the $\hat{\mathbf{z}}$ direction with the values $+1$ or -1. The magnetization, the heat capacity and the energy per particle expressions are found in Gould and Tobochnik [2010].

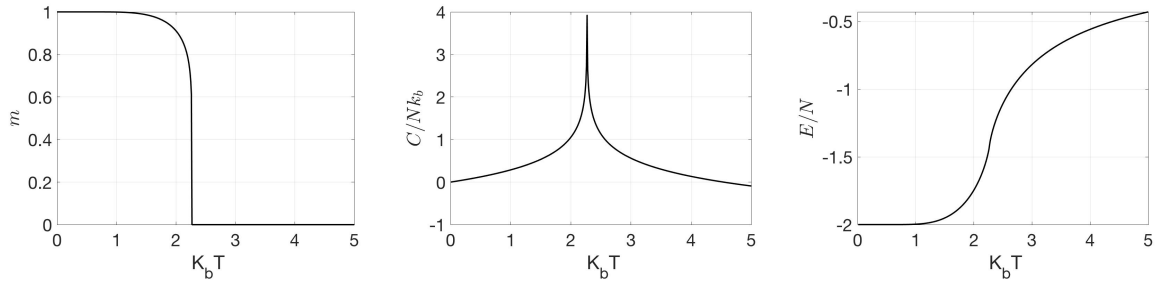
$$m(T) = \begin{cases} 0 & \text{if } T > T_c \\ \left[1 - sinh^{-4}\left(\frac{2\mathrm{J}}{kbT}\right)\right]^{\frac{1}{8}} & \text{if } T < T_c \end{cases} \tag{3.6}$$

$$\frac{C(T)}{NK_b} = -\frac{2}{\pi}\left(\frac{2\mathrm{J}}{K_b T_c}\right)^2 log\left(1 - \frac{K_b T}{K_b T_c}\right) \tag{3.7}$$

$$\frac{E(\beta)}{N} = -2\mathrm{J}tanh(2\beta\mathrm{J}) - \mathrm{J}\frac{sinh^2(2\beta\mathrm{J}) - 1}{sinh(2\beta\mathrm{J})cosh(2\beta\mathrm{J})}\left(\frac{2}{\pi}K_1(\kappa) - 1\right) \tag{3.8}$$

$$K_1(\kappa) = \int_0^{\frac{\pi}{2}} \frac{d\phi}{\sqrt{1 - \kappa^2 sin^2(\phi)}} \quad \text{and} \quad \kappa = 2\frac{sinh(2\beta\mathrm{J})}{cosh^2(2\beta\mathrm{J})} \tag{3.9}$$

Note that $\beta = 1/K_b T$. Equations 3.6, 3.7 and 3.8 are shown in the figures 3.1. The plots were computed using standard numerical tools such as matlab.



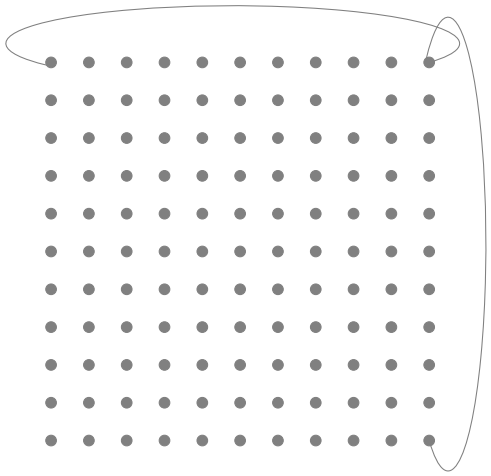
(A) (B) (C).

Figure 3.1: Analytic curve for a large number of particles in a square lattice: (A) Magnetization. (B) Specific heat capacity. (C) Energy per particle.

The analytic expressions are valid for a large number of particles in the limit that the boundary condition can be neglected. This results are essential to validate the simulation proposed in the next sections.

3.2. Metropolis algorithm, boundary condition and initial condition

The simulation must be conducted for a finite lattice. The boundary condition can be a fixed spin distribution for the surroundings of the lattice. The most typical and easy-to-implement forms are: all spins up; or all spins down for the surroundings. Another possibility is to set a periodic boundary condition in which a particle of the first column interacts with the particle of the last column (see the figure below) - the same procedure for the first and last rows.



The initial conditions can be any spin configuration. For instance, in section 3.3 the initial condition will be set as "all spins up" as well as in section 4. In appendix A.1 there is a simulation with a random-spin initialization. By implementing the Monte Carlo method's idea presented in the section 2, the algorithm 1 is proposed below.
ΔE is computed via the equation 3.3 and the probability of accepting the transition "flip the spin" is $e^{-\beta \Delta E}$. The *while* can be interpreted as a time evolution of the system and it tends to equilibrium for a large number of iterations.

Algorithm 1: Metropolis algorithm for 2D Ising model

Result: Compute spin evolution of a 2D ferromagnetic lattice;
Lattice initialization: "spin-up", "spin-down" or "random spin";
while *While equilibrium is not achieved* **do**
  Choose randomly (i,j) lattice element;
  Choose randomly r $\in [0, 1]$;
  Compute flipping energy cost $\Delta E_{i,j}$;
  if $\Delta E_{i,j} < 0$ **then**
    Flip spin (i,j);
  else if *r< exp (-*$\Delta E_{i,j}/k_bT$*)* **then**
    Flip spin (i,j);
  else
    Do not flip spin (i,j);
  end
  Compute magnetization and energy per particle;
  Draw spin lattice with physical quantities;
  end

3.3. Weak validation of algorithm 1

A strong algorithm validation would perform the *while* (algorithm 1) for a large number of iterations - and large number of particles N - for a range of temperatures from $K_bT = 0$ up to $K_bT = 5$. This procedure would allow us to reconstruct the figures 3.1(A), 3.1(B) and 3.1(C). Due to the limited access to high performance computing/clusters, in this section, it is proposed a weak algorithm validation. The validation consists in computing the physical quantities after equilibrium - for only one temperature, say T_{test} - and compare it with the analytic expressions (3.6, 3.7 and 3.8) evaluated at T_{test}. The following numerical results presented used algorithm 1 and performed 10^5 iterations.
The simulation parameters are in the table 3.1. The system evolves from a lattice with all spins

Parameter	value
Exchange interaction J	1 (ferromagnetic)
Number of lattice columns L	33
Number of particles $N = L^2$	1089
Temperature K_bT_{test}	3
Number of *while* iterations	10^5
Boundary Condition	Periodic
Initial condition	All spins up

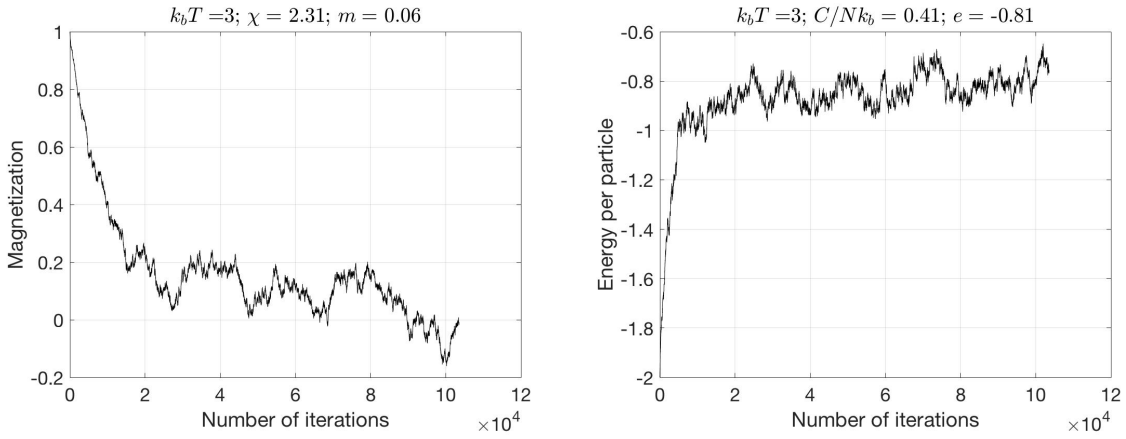
Table 3.1: Parameters for algorithm validation

up to the following final state (see figure 3.2(C)). In order to compute the physical quantities it was considered that the system achieved equilibrium from half number of iteration on - this procedure was done in order to neglect the transient regime that happens for low numbers of iterations. Since it is a 10^3-particles system, in average, each particle has had the chance to flip 10^2 times.

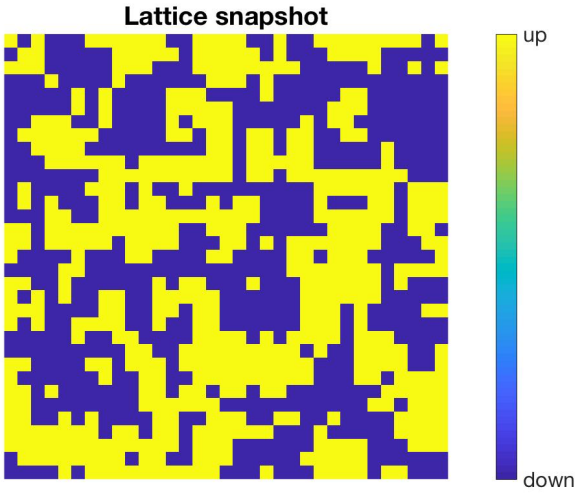
Results for $K_bT = 3$		
Physical quantity	Analytic value	Simulation
$m(T)$	0	0.06
$e(T)$	-0.8182	-0.8148
$C(T)/NK_b$	0.56	0.41

Table 3.2: Comparing analytic values (figure 3.1) and simulation (figure 3.2) .

The simulation results presented in table 3.2 are sufficiently close to the analytic values in order to infer that algorithm 1 can successfully compute the time evolution of the system - physically it is "melting" the ordered initial state to a disordered final state of zero magnetization. If the simulation were performed for a greater number of iterations it would definitely generate more accurate values for the physical quantities. However since the numerical simulation is not an infinite lattice there will be always a difference between the analytical values and the simulation results due to the boundary condition effect. Therefore, this simulation basically shows that the Ising model is capable of modelling the order-disorder transition.
It is remarkable that even if the mean magnetization, computed via simulation, is very close to zero, the snapshot of figure 3.2(C) shows a certain correlation between spins. The image is sometimes called "lake" of spins. There is an additional simulation in the appendix A.1 to demonstrate it.



(A) (B)



(C).

Figure 3.2: Simulation results with parameters of table 3.1. (A) Magnetization with transient regime. (B) Energy per particle with transient regime. (C) Lattice snapshot after 10^5 iterations.

4. Investigation of finite lattice proprieties at low temperature

This section aims to investigate the relationship between the 2D-lattice side L and the physical quantities such as the magnetization, energy per particle and specific heat capacity. For that, the algorithm 1 was performed using periodic boundary condition for different Ls at low temperature - below the critical temperature.

A proper numerical simulation would be a sweep of temperature range from $K_bT = 0$ up to $K_bT = 5$ and a sweep of L from $L = 2$ up to $L = 10^3$.

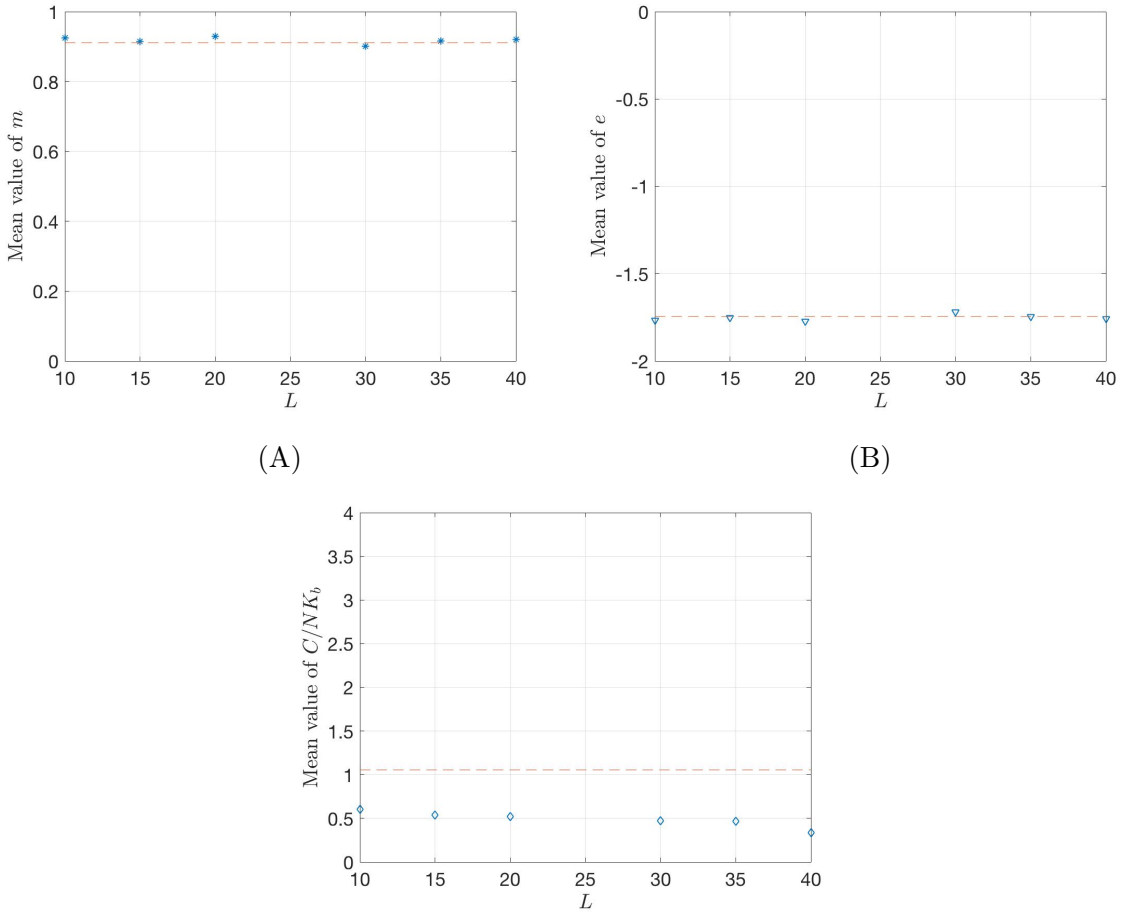
Again, due to limited access to high performance computing a simpler case will be studied at a fixed temperature.

The most natural temperature to be chosen in order to perform this numerical experiment would be the critical temperature. However the system at the critical temperature exhibit a heat capacity peak meaning the variance of the magnetization to be large. The consequence of that is a system that takes a lot of iterations to achieve equilibrium. That being said, the chosen temperature to run the numerical simulation will be $K_BT = 2$ in order to be less computationally demanding.

If the all-spins-up initial state is set, the energy per particle associated to it is $e = -2$ whereas the energy per particle of the random-spins initial state is $e = 0$. From the energy perspective, it can be easily verified in figure 3.1(C) that the shortest way to achieve the equilibrium at $K_B T = 2$ is to set all-spins-up as initial condition. This choice also demands less computational effort to achieve equilibrium. The parameters of the simulation are in the table 4.1.

Parameter	value
Exchange interaction J	1 (ferromagnetic)
Number of lattice columns L	10,15,20,30,35,40
Temperature $K_b T_{test}$	2
Number of *while* iterations	$10^4 - 10^5$
Boundary Condition	Periodic
Initial condition	All spins up

Table 4.1: Simulation parameters in order to investigate the dependence of physical quantities on L.



(A) (B)

(C).

Figure 4.1: Investigation of physical quantities L. - - - - - Analytic physical quantity at $K_b T = 2$. (A) Magnetization. (B) Energy per particle. (C) Specific heat capacity.

In general, the numerical results of mean magnetization and mean energy per particle are approximately the same as the analytic values. In appendix A.2 it is notable that for low L the variance of the energy per particle and the magnetization increases drastically.

Figure 4.1(C) shows that the specific heat capacity decreases for larger L. See appendix A.2 to visualize the variance of m and e for $L = 10$ to $L = 40$.

5. Conclusion

The Monte Carlo method is an extensive topic and has numerous applications in physics and other fields. The results here presented are a simple demonstration of Monte Carlo method's robustness using the framework of Metropolis algorithm. The 2D Ising model is an useful case that allow us validate numerical simulation thanks to its analytical results.
In section 3.3 the Metropolis algorithm has shown its potential providing a phase transition behavior similarly to the analytic results from section 3.1.
In section 4 it is shown - at low temperatures - how the physical quantities vary with respect to L: the side o the two-dimensional square lattice.
In appendix B there is the matlab scripts used in order to perform the numerical simulations here presented. This project has also shown that Monte Carlo simulations have a very demanding computational cost even for simple cases such as the two-dimensional Ising model. If high performance computers are available for research, more complicated and realistic cases can be studied via Monte Carlo simulation.

References

Harvey Gould and Jan Tobochnik. *Statistical and thermal physics: with computer applications*. Princeton University Press, 2010.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Lars Onsager. Crystal statistics. i. a two-dimensional model with an order-disorder transition. *Physical Review*, 65(3-4):117, 1944.

Silvio Salinas. *Introduction to statistical physics*. Springer Science & Business Media, 2001.

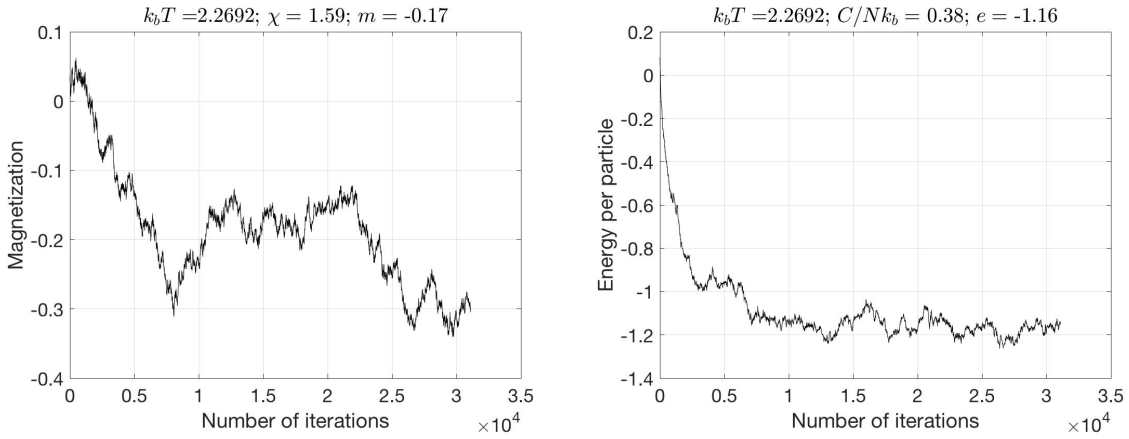
Chen Ning Yang. The spontaneous magnetization of a two-dimensional ising model. *Physical Review*, 85(5):808, 1952.

A. Additional numerical simulations

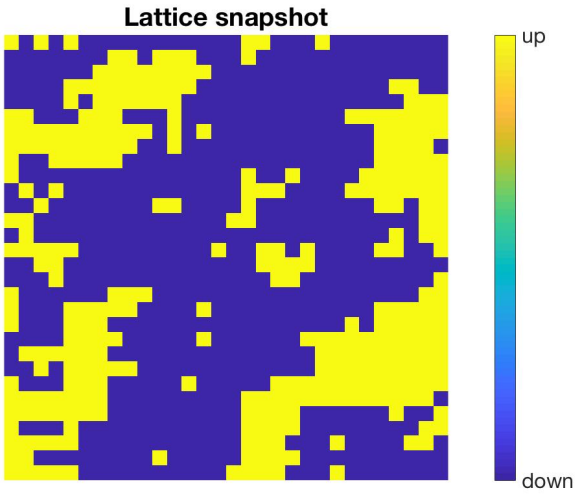
A.1. Illustration: lake of spins

Parameter	value
Exchange interaction J	1 (ferromagnetic)
Number of lattice columns L	30
Temperature K_bT	K_bT_c
Number of *while* iterations	3×10^4
Boundary Condition	Periodic
Initial condition	Random spins

Table A.1: Lake of spins demonstration



(A) (B)



(C).

Figure A.1: Lake-of-spins simulation result with parameters of table A.1. (A) Magnetization with transient regime. (B) Energy per particle with transient regime. (C) Lattice snapshot after 4×10^4 iterations.

A.2. $L = 10$ and $L = 40$ simulations

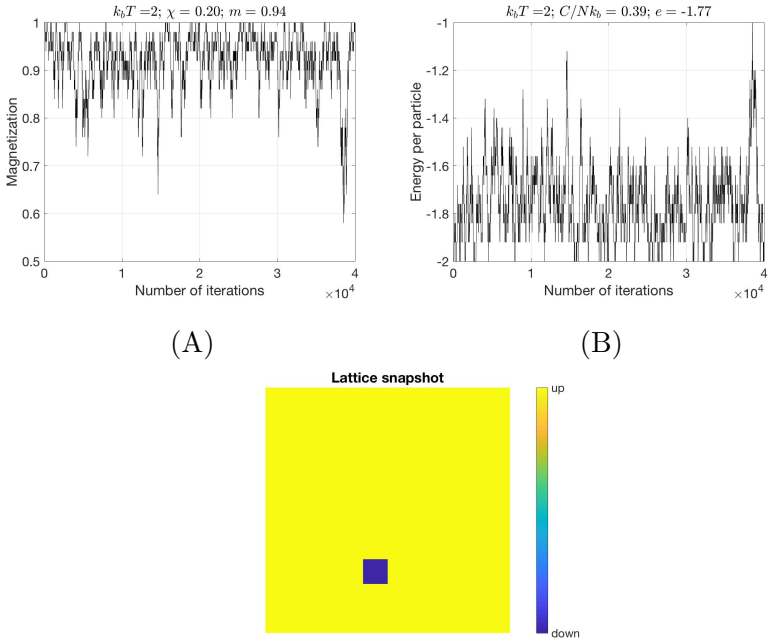


Figure A.2: $L = 10$ Simulation result with parameters of table 4.1. (A) Magnetization with transient regime. (B) Energy per particle with transient regime. (C) Lattice snapshot after 4×10^4 iterations.

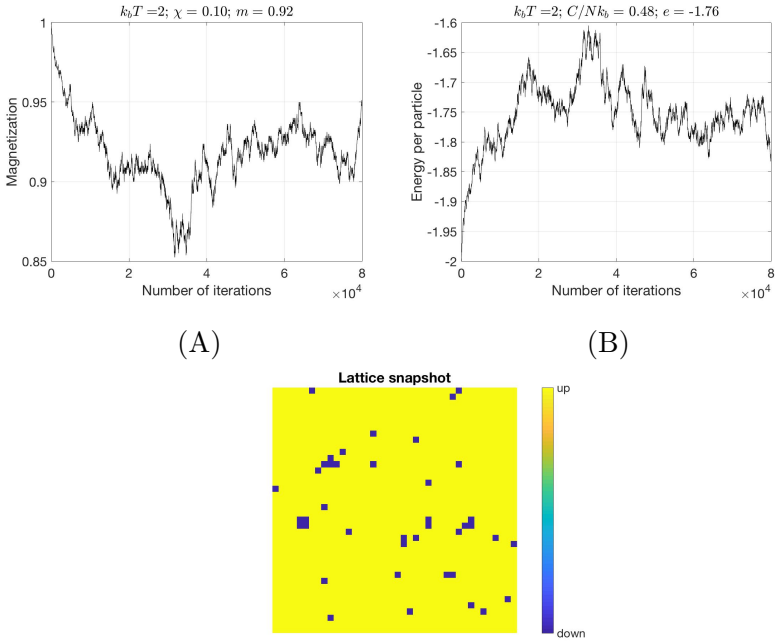


Figure A.3: $L = 40$ Simulation result with parameters of table 4.1. (A) Magnetization with transient regime. (B) Energy per particle with transient regime. (C) Lattice snapshot after 4×10^4 iterations.

B. Matlab script

B.1. Metropolis algorithm

```
function [m,e,s,parameters] = isingmodel(J,L,kbT,Ne,BC,IC,Display,...
                                                    m_prev,e_prev,
                                                      s_prev)
    %% Description 25/07/2020 (Renan Liupekevicius Carnielli, ra
       157139)

    % This script computes the spin flips a certain amount of times
       for
    %  a 2D square with periodic boundary conditions or fixes spin-up
    % boundary condition.
    % There is no external magnetic field applied to the system.

    % RETURN:
    % m  - magnetization evolution
    % e  - energy per particle evolution
    % s -  spin matrix (=lattice)
    % parameters  - Input

    % %% Parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % J   ferromagnetic (>0) exchange interaction
    % L   number of particles on an edge
    % kbT Boltzmant contant Kb times temperature
    % Ne  Number of iteractions (hopefully) sufficient to achive
       equil.
    % BC  Boundary Condition "periodic" or "up"
    % IC  Inicial Condition "up", "down", "rand" or "previous
    % Display = "1" plot or "0" do not plot

    %m_prev,e_prev,s_prev

    %% Fixed parameters
    % N       = L^2;                          %number of particles
    % m(T) = | 0                             if T > Tc
    %        |(1-sinh(2J/kbT)^(-4))^(1/8)     if T < Tc

    %% Initialize
    tic

    % Magnetization and energy per particle
    m=[];
    e=[];

    % Initialize spin lattice
    switch IC
        case "up"   % Initialize spin-up lattice
            s   = ones(L,L, Ne);

        case "down" % Initialize spin down
```

```
45              s    = -ones(L,L, Ne);
46
47          case "rand" % Initialize random spin
48              randspin = 2*randi([0 1],L) - ones(L,L);
49              s        = repmat(randspin,1,1,Ne);
50
51          case "previous" % Initialize with previous computed lattice
52              % Number of input arguments
53              switch nargin
54
55                  case 10
56                  m = m_prev;
57                  e = e_prev;
58                  s = repmat(s_prev(:,:,end),1,1,Ne);
59                  shape=size(s);
60                  Ntotal = shape(3) + Ne;
61
62                  otherwise
63                  error("Input arguments error."+...
64                      " Try another initial condition IC.")
65              end
66          otherwise
67                  error("Choose the initial conditions 'up',"+...
68                      " 'down', 'rand' or 'previous'")
69
70       end
71
72
73      % Indices randomly choosen
74       randindex = randi(L,[2,Ne]);   % Random indices matrix
75
76
77      % Random probability of flipping that is be compared to exp(-DE/
          kbT)
78       prob      = rand(Ne,1);
79
80
81
82       % Initialize s_prev in case IC is different than "previous"
83       if IC ~= "previous"
84           s_prev=[];
85       end
86
87
88
89
90       %% Compute flips (hopefully) until equilibrium
91
92  switch Display
93
94       case 1
```

```
95              for k=2:Ne
96                  % Random index (i,j)
97                  i= randindex(1,k); j = randindex(2,k);
98
99                  % Compute Delta E and flip/not flip
100                 DE = ComputeDeltaE(J,s,i,j,k,L,BC);
101                 if DE < 0
102                     s(i,j,k:end)=-s(i,j,k-1); % flip
103                 elseif prob(k) < exp(- DE / kbT)
104                     s(i,j,k:end)=-s(i,j,k-1); % flip
105                 else
106                     % not flip
107                 end
108
109                 % Magnetization and Energy per particle evolution
110                 mk = Compute_m(s,k,L^2);
111                 ek = Compute_E(s,k,J,L,BC);
112
113                 % Magnetization and Energy per particle storage
114                 m = [m mk];
115                 e = [e ek];
116
117                 % Selecting 3 digits for display
118                 mag                 = num2str(mk,'%.3f');
119                 energyperparticle = num2str(ek,'%.3f');
120
121                 % Draw lattice
122                 titre="Computed "+ num2str(100*k/Ne,'%.f')+"%;"+...
123                           " m = "+ mag+...
124                          "; e = "+ energyperparticle + ...
125                          "; kbT= "+ num2str(kbT,'%.2f');
126
127                 imagesc(s(:,:,k));
128                 set(gca,'fontsize', 18);
129                 title(titre);
130                 colorbar('Ticks',[-1,1],'TickLabels',{'down','up'})
131                 axis equal off;
132                 drawnow;
133
134             end
135
136     case 0
137             for k=2:Ne
138
139                 % Random index (i,j)
140                 i= randindex(1,k); j = randindex(2,k);
141
142                 % Compute Delta E and flip/not flip
143                 DE = ComputeDeltaE(J,s,i,j,k,L,"periodic");
144                 if DE < 0
145                     s(i,j,k:end)=-s(i,j,k-1); % flip
```

```
146                     elseif prob(k) < exp(- DE / kbT)
147                         s(i,j,k:end)=-s(i,j,k-1); % flip
148                     else
149                         % not flip
150                     end
151
152                     % Magnetization and Energy per particle evolution
153                     mk = Compute_m(s,k,L^2);
154                     ek = Compute_E(s,k,J,L,"periodic");
155
156                     % Magnetization and Energy per particle storage
157                     m = [m mk];
158                     e = [e ek];
159                 end
160 end
161
162
163     %% return
164
165     % Concatenate s
166     if IC=="previous"
167      s = cat(3,s_prev,s);
168      shape=size(s);
169      Ntotal = shape(3);
170     else
171         Ntotal=Ne;
172     end
173
174 %     % Save variables
175 %     filename= w +'.mat';
176 %     save(filename,'m','e','s','parameters');
177
178     % Save parameters of current execution
179     parameters = struct( "L", L,"J",J,"kbT",kbT,...
180                          "Ne",Ntotal,"BC",BC,"IC",IC);
181
182
183
184     %Display message
185     disp(" ising model has been succesfully computed");
186
187     toc
188 end
189
190 function DE = ComputeDeltaE(J,s,i,j,k,l, BC)
191 %% Description
192 %
193 % Renan Liupekevicius Carnielli, ra 157139, 25/07/2020
194 %
195 % This function computes de energy variation from spin s to -s in a
196 % position (i,j) of a square lattice with periodic bounday condition.
```

```
197 %
198 % J - Exchange iteraction;
199 % S - Spin matrix that represents the lattice;
200 % i,j - Coordinates of the potential flipping spin;
201 % k - A snapshot of the system that will evolve to state
202 % k+1 by flipping a spin or not.
203 %
204 % shape = size(s);
205 % l     = shape(1);
206 %
207 %% Equation
208 % Initial first neighbors energy (before flip):
209 % Ei       = -J s    ( s_i+1j + s_i-1j + s_ij+1 +  s_ij-1 )
210 %
211 % Final first neighbors energy (after flip):
212 % Ef       = -J (-s) ( s_i+1j + s_i-1j + s_ij+1 +  s_ij-1 )
213 %
214 % Energy variation of the whole lattice is given by flipping s:
215 %
216 % DeltaE   = Ef - Ei
217 %          = 2Js ( s_i+1j + s_i-1j + s_ij+1 +  s_ij-1 )
218 %
219 %% Compute Delta E
220
221 switch BC
222
223     case "periodic"
224         % first lattice row (upper horizontal edge)
225         if i == 1
226             if j == 1   % first element (upper left corner)
227                 DE =  s(i+1,j,k)+ s( l ,j,k) + s(i,j+1,k) + s(i, l ,k
                        );
228             elseif j==l % last element  (upper right corner)
229                 DE =  s(i+1,j,k)+ s( l ,j,k) + s(i, 1 ,k) + s(i,j-1,k
                        );
230             else        % middle elements
231                 DE =  s(i+1,j,k)+ s( l ,j,k) + s(i,j+1,k) + s(i,j-1,k
                        );
232             end
233
234
235
236         % last lattice row (lower horizontal edge)
237         elseif i == l
238             if j == 1   % first element (lower left corner)
239                 DE = s( 1 ,j,k)+ s(i-1,j,k) + s(i,j+1,k) + s(i, l ,k)
                        ;
240             elseif j==l % last element (lower right corner)
241                 DE = s( 1 ,j,k)+ s(i-1,j,k) + s(i, 1 ,k) + s(i,j-1,k)
                        ;
242             else        % middle elements
```

```
243                 DE = s( 1 ,j,k)+ s(i-1,j,k) + s(i,j+1,k) + s(i,j-1,k)
                        ;
244             end
245
246
247
248         % vertical edges and middle
249         elseif j == 1     % first lattice column (middle elements)
250                 DE = s(i+1,j,k)+ s(i-1,j,k) + s(i,j+1,k) + s(i, l ,k)
                        ;
251
252
253
254         elseif j == l % last lattice column (middle elements)
255                 DE  = s(i+1,j,k)+ s(i-1,j,k) + s(i, 1 ,k) + s(i,j-1,k
                    ) ;
256
257
258
259         else           % middle elements
260                 DE  = s(i+1,j,k)+ s(i-1,j,k) + s(i,j+1,k) + s(i,j-1,k
                    ) ;
261
262         end
263
264     case "up"
265
266         % first lattice row (upper horizontal edge)
267         if i == 1
268             if j == 1   % first element (upper left corner)
269                 DE =  s(i+1,j,k)+      1     + s(i,j+1,k) +     1
                          ;
270             elseif j==l % last element  (upper right corner)
271                 DE =  s(i+1,j,k)+      1     +      1     + s(i,j-1,k
                        );
272             else        % middle elements (upper horizontal edge)
273                 DE =  s(i+1,j,k)+      1     + s(i,j+1,k) + s(i,j-1,k
                        );
274             end
275
276
277
278         % last lattice row (lower horizontal edge)
279         elseif i == l
280             if j == 1   % first element (lower left corner)
281                 DE =      1    + s(i-1,j,k) + s(i,j+1,k) +      1
                        ;
282             elseif j==l % last element (lower right corner)
283                 DE =      1    + s(i-1,j,k) +      1     + s(i,j-1,k)
                        ;
284             else        % middle elements (lower horizontal edge)
```

```
285                 DE =       1      + s(i-1,j,k) + s(i,j+1,k) + s(i,j-1,k
                        );
286             end
287
288
289
290         % vertical edges and middle
291         elseif j == 1     % first lattice column (middle elements)
292                 DE  = s(i+1,j,k)+ s(i-1,j,k) + s(i,j+1,k) +     1     ;
293
294
295
296         elseif j == l % last lattice column (middle elements)
297                 DE  = s(i+1,j,k)+ s(i-1,j,k) +     1       + s(i,j-1,k)
                        ;
298
299
300
301         else            % middle elements
302                 DE  = s(i+1,j,k)+ s(i-1,j,k) + s(i,j+1,k) + s(i,j-1,k
                       ) ;
303         end
304
305 end
306 %% Return
307 DE = 2*J*s(i,j,k)*DE;
308
309 end
310
311
312 function E = Compute_E(s,k,J,L,BC)
313 %% Description
314 % Compute the energy with no external magnetic field using Ising's
       model.
315
316 switch BC
317
318     case "periodic"
319
320     % Edges (periodic BC)
321     E = dot(s(1,:,k),s(L,:,k)) + dot(s(:,1,k),s(:,L,k));
322
323     % Middle
324     for i=1:L-1
325         E = E + dot(s(i,:,k),s(i+1,:,k)) + dot(s(:,i,k),s(:,i+1,k));
326     end
327
328     case "up"
329
330     E = sum(s(1,:,k)) + sum(s(L,:,k)) + sum(s(:,1,k)) + sum(s(:,L,k))
           ;
```

```
331
332         for  i=2:L-2
333
334             E = E + dot(s(i,:,k),s(i+1,:,k)) + dot(s(:,i,k),s(:,i+1,k));
335         end
336
337 end
338
339 E = -J*E/L^2;
340 end
341
342
343 function m = Compute_m(s,k,N)
344 %% Description
345 % s - is a 2D-lattice of spins 1 or -1
346 % L - is an integer that represents the number of particles on an
        edge
347
348  m = sum(s(:,:,k),"all") / N;
349
350 end
```

B.2. Plots

```
% clear all;
 close all;


%% Description 26/07/2020 (Renan Liupekevicius Carnielli, ra 157139)

% This is an assignment of F604 Statistical Physics.
% University of Campinas.


% This script displays the behavior of an evolving 2D square lattice.
% Each cell has a spin up or a spin down. There is an interaction
% between the spins using Ising Model.

%% Constants
kb   = 1.380649e-23;              % Boltzmann Constant
kbTc = 2/log(1+sqrt(2));       % Critical temperature (2D) times Kb


%% Parameters
J    = 1;       % ferromagnetic (>0) exchange interaction
L    = 20;     % number of particles on an edge
N    = L^2;    % number of particles
kbT = 2;       % Boltzmant contant Kb times temperature
Ne   = 100;  % Number of flips
BC   = "periodic"; % Boundary Condition "periodic" or "up"
IC   = "previous";   % Inicial Condition "up", "down", "rand" or "
    previous"


%% Compute flips via metropolis algorithm
switch IC
    case "previous"

     % Parameters
     J    = parameters.J;
     L    = parameters.L;
     N    = L^2;
     kbT = parameters.kbT;
     BC   = parameters.BC ;

     % Continue simulation
     [m,e,s,parameters] = isingmodel(J,L,kbT,Ne,BC,IC,1,m,e,s);

    otherwise
     % New simulation
     [m,e,s,parameters] = isingmodel(J,L,kbT,Ne,BC,IC,1);
end
```

```
50
51
52
53 %% Plot: magnetizaion, energy, suceptibilty and heat capacity
54
55  % Set time-step Window to compute moving average
56  fenetre = 10000;
57
58  % Compute physical quantities from 'initial_index' until the end of
       the
59  % vector
60  initial_index =round(length(m)/2);
61
62  % Compute magnetization (hopefully after equilibrium)
63  m_mean  = mean(m(initial_index:end));
64
65  % Compute energy per particle (hopefully after equilibrium)
66  e_mean  = mean(e(initial_index:end));
67
68  % Compute magnetization variance: mag. suceptibility
69  chi     = L^2 * 1/kbT     * var(m(initial_index:end));
70
71  % Compute energy variance: Heat Capacity over kb:   C/kb^
72  CokboN    = L^2 * 1/(kbT^2) * var(e(initial_index:end));
73
74  % Magnetization plot
75  figure(2)
76  hold on;
77  plot(m,'k');
78  %plot(movmean(m,fenetre))
79  box on;
80  grid on;
81  set(gca,'fontsize', 18);
82  xlabel('Number of iterations');
83  ylabel('Magnetization');
84  title("$k_bT= $"+num2str(kbT)+...
85        ";  $\chi =$ "+num2str(chi,'   %.2f')+ ...
86        ";  $m    =$ "+num2str(m_mean,'%.2f'), ...
87                              'Interpreter','latex');
88  hold off;
89
90  %Energy per particle plot
91  figure(3)
92  hold on;
93  plot(e,'k');
94  %plot(movmean(e,fenetre))
95  box on;
96  grid on;
97  set(gca,'fontsize', 18);
98  xlabel('Number of iterations');
99  ylabel('Energy per particle');
```

```
100 title("$k_bT= $"+num2str(kbT)+...
101       "; $C/N k_b =$ "+num2str(CokboN,'%.2f')+ ...
102       "; $e =$ "+num2str(e_mean, '%.2f'), ...
103                                 'Interpreter','latex');
104 hold off;
105
106
107 % Final state (snapshot) plot
108 figure(7)
109 titre="Lattice snapshot";
110 imagesc(s(:,:,end));
111 set(gca,'fontsize', 18);
112 title(titre);
113 colorbar('Ticks',[-1,1],'TickLabels',{'down','up'})
114 axis equal off;
115 drawnow;
116
117
118
119
120 %% Store workspace file
121
122 % set file name
123 filename = "parameters_L"+ parameters.L +"_kbT"+ parameters.kbT +...
124            "_Ne"+ parameters.Ne+ "_BC"+ parameters.BC;
125
126 %save the workspace as matlab file
127 save(filename+".mat", '-v7.3');
128
129
130
131
132 %% Analytic expressions
133
134    % temperature vector and critical temperature
135    kbT_vec    = 0:0.01:5;              % Temperature times Kb (vector
       )
136
137
138    % Analytic magnectization of 2D ising model
139    ma  = [(1-sinh(2*J./kbT_vec(1:227)).^(-4)).^(1/8) zeros
       (1,501-227)];
140
141
142    % Analytic heat capacity
143    C_okb_oN_analytic = - 2/pi * (2*J/kbTc)^2* log (abs(1-kbT_vec/
       kbTc));
144
145
146    % Analytic energy per particle
147    beta  = 1./kbT_vec;
```

```
148     kappa = 2 * ( sinh(2 * beta *J) ) ./ ( cosh(2 * beta  * J) ).^2 ;
149
150     K1    = zeros(1,length(beta));
151     x     = 0:0.01:pi/2;
152
153     for i= 1:length(beta)
154     fun       = @(x)  1 ./ (     sqrt( 1- kappa(i)^2 * (sin(x)).^2 )
            );
155     K1(i)     = trapz(x,fun(x));
156     end
157
158     e_analytic = -2*J* tanh( 2 * beta * J) ...
159                  - J * ( sinh(2 *beta * J).^2 -1 )...
160                  ./( sinh(2 * beta * J) .* cosh(2 * beta * J) ) ...
161                  .* ( 2 / pi .* K1 -1 );
162
163 %% Plot analytic expressions
164
165     % Plot analytic magnetization
166     figure(4)
167     hold on;
168     box on;
169     plot(kbT_vec,ma,'k','LineWidth',2);
170    % plot(kbT_vec,ones(size(kbT_vec)),'k--','LineWidth',2);
171     grid on;
172     set(gca,'fontsize', 25);
173     xlabel(' K_bT');
174     ylabel('$m$', 'interpreter','latex');
175     hold off;
176
177
178     %Plot analytic heat capacity
179     figure(5)
180     hold on;
181     box on;
182     plot(kbT_vec,C_okb_oN_analytic,'k','LineWidth',2);
183     grid on;
184     set(gca,'fontsize', 25);
185     xlabel(' K_bT');
186     ylabel('$C/N k_b$', 'interpreter','latex');
187     hold off;
188
189
190     %Plot analytic energy per particle
191     figure(6)
192     hold on;
193     box on;
194     plot(kbT_vec,e_analytic,'k','LineWidth',2);
195     grid on;
196     set(gca,'fontsize', 25);
197     xlabel(' K_bT');
```

```
198     ylabel('$E/N $', 'interpreter','latex');
199     hold off;
```